

typically uses one of two known approaches: translation or interpretation. Translation converts a program in code native to the target machine to a program in code native to the host machine (application page 2, lines 13-16). Translation may be static or dynamic. Static translation translates all of the program code of the target machine prior to any execution of the translated code file (application page 5, lines 21-22). Dynamic translation generates code native to the host machine "on the fly." Translation generally is complex, especially when the instruction sets of the target and host machines are significantly different (application page 2, lines 19-20). An interpreter, on the other hand, "interprets" each instruction of a target program and then performs equivalent operations on the host machine, without requiring translation of the code. Interpretive execution is typically much slower than the execution of native code generated through translation (application page 3, lines 1-2).

The present invention takes a novel approach to emulation that overcomes the shortcomings of conventional emulation. According to the present invention, the target program to be emulated (*e.g.*, a program comprising instructions of the Unisys "E-mode" instruction set) is first *statically translated* to a series of instructions of an *intermediate instruction set*. The intermediate instruction set is an *instruction set that is optimized for interpretation* on the host computer (*e.g.*, a computer that employs the Alpha microprocessor). The series of translated instructions is then *executed by interpretation* on the host computer (*e.g.*, an Alpha server). The intermediate instruction set is not the native instruction set of either the target program (*e.g.*, "E-mode" instruction set) or the host computer (*e.g.*, Alpha microprocessor instruction set). Because the intermediate instruction set is optimized for interpretation on the host computer, the execution of the statically translated intermediate code by interpretation on the host computer will generally be faster than conventional interpretation.

The present invention, as recited in claims 1, 9, 19, and 27, includes features that are not disclosed or suggested by the cited references either alone or in combination. The features are represented by claim 1:

1. A method for emulating the execution of a target program comprising instructions of an instruction set of a target on a host computer *having a different instruction set*, said method comprising:  
performing a *static translation* of the instructions of the target program into a series of instructions of an intermediate instruction set, the *intermediate instruction set being optimized for*

*interpretation* on the host computer; and

*executing the series of instructions of the intermediate instruction set by interpretation* on the host computer. (emphasis added)

Horwat is directed to translating program code so that it may be executed on another machine having a different architecture. More particularly, Horwat focuses on only one aspect of such translation, namely endian format translation that provides endian-independent representations of data, pointer data, operands, and pointer operands. That is, some machines may be big-endian and other machines may be little-endian. Big-endian format maps the lowest address to the highest order data byte and little-endian format maps the lowest address to the lowest order data byte. When translating between machines having different endian formats, bytes of each instruction or each piece of data may be reordered for the architecture of a second machine. Horwat discloses such reordering of bytes.

Horwat does not disclose or suggest the invention as claimed. First, Horwat only discloses *translation* as the second step in running a program (Horwat Figure 2). That is, Horwat does not disclose *executing instructions by interpretation*. On the contrary, Horwat notes that Intercode object code is provided and therefore the object code can be run with an Intercode *translator* (Horwat col. 8, lines 4-14). *Translation*, however, is very different from *interpretation*, as described in more detail above and as acknowledged in the background of Horwat (Horwat col. 1, lines 22-49). Therefore, Horwat does not disclose *executing instructions by interpretation*, as recited by the claims.

Nor does Horwat disclose *an intermediate instruction set that is optimized for interpretation*; rather, Horwat merely discloses that during translation, performance-critical sections of a program can be *optimized for a particular target architecture* (Horwat col. 7, lines 7-12). Disclosing that sections of a program can be optimized during translation is not the same as statically translating a target program into instructions of an *intermediate instruction set that is optimized for interpretation*. Therefore, Horwat does not disclose *an intermediate instruction set that is optimized for interpretation*, as recited by the claims.

Moreover, Horwat does not disclose anything about *different instruction sets*; rather, Horwat describes how bytes can be rearranged to execute on another machine having a *different endian*

*format*. As stated in the present application, translation generally is complex, especially when the instruction sets of the target and host machines are significantly different (application page 2, lines 19-20). Therefore, Horwat does not disclose *different instruction sets*, as recited by the claims.

Further, Horwat discloses generating an object code format that is *translated at run time* (Horwat column 3, lines 2-5) and therefore discloses *dynamic translation*. As such, Horwat does not disclose *static translation*, as recited by the claims.

Shah is directed to improving the speed of dynamic translation systems by efficiently positioning translated instructions in a memory. Inefficient jumping may be reduced by positioning sections of translated instructions with respect to each other and with respect to specialized instructions (*e.g.*, trampoline instructions) that redirect control flow.

The Examiner states that Shah provides the motivation to perform static translation by noting that one advantage of static translation is that it provides the ability to more easily perform optimization during translation. (Shah column 2, lines 6-10). Shah, however, does not disclose anything about *executing instructions by interpretation*. Thus, combining the teachings of Shah with the teachings of Horwat would not result in the present invention as claimed.

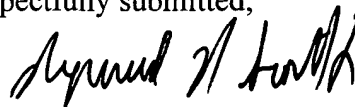
For the foregoing reasons, Applicants respectfully request reconsideration of the Section 103(a) rejection of claims 1-31.

### CONCLUSION

For the foregoing reasons, Applicants respectfully submit that all of the claims of the present application patentably define over the cited references of record, alone or in combination. Reconsideration of the Office Action and an early Notice of Allowance are respectfully requested. In the event that the Examiner cannot allow the present application for any reason, the Examiner is encouraged to contact the undersigned attorney, Raymond N. Scott Jr. at (215) 564-8951, to discuss resolution of any remaining issues.

Date: 6/25/02

Respectfully submitted,



Raymond N. Scott Jr.  
Attorney for Applicant  
Registration No. 48,666

WOODCOCK WASHBURN LLP  
One Liberty Place - 46<sup>th</sup> Floor  
Philadelphia, Pennsylvania 19103  
Telephone: 215.568.3100  
Facsimile: 215.568.3439